Trucchi del mestiere

Una bella rubrica del Mathematica Journal *è intitolata* Tricks of the Trade *(trucchi del mestiere) e mostra una serie di problemi che si presentano spesso agli utenti inesperti suggerendo le relative soluzioni eleganti e/o involute. Ispirandomi liberamente a tale rubrica cerco di rispondere da queste colonne alle domande ricevute via e-mail da quando scrivo su MC, presentando alcuni esempi, che permettono di evidenziare un po' per volta alcuni dei trucchi più "sporchi" che ho imparato negli ultimi anni*

di Francesco Romani

Predicati

Le espressioni logiche in *Mathematica* possono avere tre valori: vero (**True**), falso (**False**) e indeterminato. Un esempio di quest'ultima possibilità si ha nel confronto tra un numero e una variabile simbolica.

In[1]:=
```
Pi==3.14
```
Out[1]=
```
Pi == 3.14
```
Se un'espressione indeterminata come questa è usata in un **If** con una o due alternative anche l'**If** rimane indeterminato

In[2]:=
```
If[Pi==3.14,"sì"]
```
Out[2]=
```
If[Pi == 3.14, sì]
```
In[3]:=
```
If[Pi==3.14,"sì","no"]
```
Out[3]=
```
If[Pi == 3.14, sì, no]
```
Se l'**If** ha tre alternative viene valutata la terza.

In[4]:=
```
If[Pi==3.14,"sì","no","forse"]
```
Out[4]=
```
forse
```
Nell'esempio seguente il valore dell'**If** viene assegnato ad una variabile il cui valore calcolato dipende dall'espressione logica. In questo caso si comprende lo scopo di tutto il marchingegno: ritardare la valutazione dell'espressione al momento in cui essa diviene determinata.

In[5]:=
```
b=If[a==1,"sì","no"]
```
Out[5]=
```
If[a == 1, sì, no]
```
In[6]:=
```
a=1;
b
```
Out[6]=
```
sì
```
In[7]:=
```
a=0;
b
```
Out[7]=
```
no
```
In[8]:=
```
Clear[a]
b
```
Out[8]=
```
If[a == 1, sì, no]
```
Si può forzare la valutazione di un'espressione logica con i predicati

TrueQ[x], che vale **False** se **x** non vale **True**.

SameQ[x,y], che si abbrevia in **x===y** e vale **True** se l'espressione **x** è identica a **y** e **False** altrimenti.

In[9]:=
```
TrueQ[Pi==3.14]
Pi===3.14
```
Out[9]=
```
False
False
```
Un altro metodo, che funziona quando si ha a che fare con variabili che possono essere valutate numericamente, è quello di usare la Funzione **N[]**. Attenzione però a tenere conto degli effetti della rappresentazione in aritmetica finita: **N[Pi]** è (sul Mac) un numero di macchina con circa 19 cifre esatte.

In[10]:=
```
InputForm[N[Pi]]
```
Out[10]=
```
3.141592653589793238
```
Ovviamente **N[Pi]** non è uguale a 3.14.

In[11]:=
```
N[Pi]==3.14
```
Out[11]=
```
False
```
Ma la loro differenza è più piccola di 0.1

In[12]:=
```
Abs[N[Pi]-3.14]<0.1
```
Out[12]=
```
True
```
Questa seconda forma di confronto è utile quando si devono confrontare due numeri a meno di una prestabilita tolleranza. Si noti che per numeri di macchina molto vicini tra loro **Equal[x,y]** (ovvero **x==y**) è meno esigente di **SameQ[x,y]** (ovvero **x===y**).

In[13]:=
```
1.0000000000000001 ==
1.0000000000000002
```
Out[13]=
```
False
```
In[14]:=

```
1.00000000000000001 ==
1.00000000000000002
Out[14]=
True
In[15]:=
1.00000000000000001 ===
1.00000000000000002
Out[15]=
False
In[16]:=
1.000000000000000000000001 ===
1.000000000000000000000002
Out[16]=
True
```

Una mancata considerazione di questi fenomeni può provocare pasticci colossali in programmi particolarmente critici.

HoldForm

Utilizzando le definizioni ricorsive si può costruire una funzione con una crescita spaventosa. La funzione `f[x,y,z]` per `z`=1 è la semplice addizione; se il secondo argomento è 1 e il terzo è maggiore di 1 il risultato è x; nel caso generale, `f[x,y,z]` itera `y` volte l'operazione di livello `z-1` su `x`.

```
In[1]:=
f[x_,y_,1 ]:=x+y;
f[x_,1 ,z_]:=x/;x>1;
f[x_,y_,z_]:=f[x,f[x,y-1,z],z-1];
```

Detto così sembra difficile ma se si guarda meglio si vede che `f[x,y,2]` definisce la moltiplicazione, `f[x,y,3]` definisce l'elevamento a potenza, `f[x,y,4]` è la torre di potenze `x^(x^(x...)))` e così via.

```
In[2]:=
f[3,3,1]
f[3,3,2]
f[3,3,3]
```

Mathematica su PowerPC: è arrivata !!

Riprendiamo il discorso della velocità di Mathematica *sulle varie piattaforme con ospiti di eccezione i PowerMacintosh che montano il microprocessore RISC PowerPC frutto della collaborazione tra Apple, IBM e Motorola. (Per i dettagli si vedano i molti articoli apparsi su MC e in particolare la presentazione, a cura di Andrea de Prisco, sul numero di marzo e l'esauriente prova del PowerMacintosh pubblicata in questo numero).*

L'occasione

Il 10 marzo la Apple ha presentato le nuove macchine RISC PowerMacintosh, il 17 marzo il Presidente della Wolfram-Europe, Conrad Wolfram è venuto in Italia per presentare (presso la Scuola Normale Superiore di Pisa e l'Università di Milano) le potenzialità della nuova versione di *Mathematica*, capace di girare in modo nativo sui nuovi PowerMacintosh. Coinvolto nell'evento (ho fatto una chiacchierata di mezz'ora sulle possibilità di *Mathematica* come strumento di comunicazione didattica e scientifica) ho approfittato dell'occasione per girare il mio solito benchmark su tutte le configurazioni interessanti che sono riuscito a raggiungere.

Mathematica *sul Mac*

Con la distribuzione della versione nativa di *Mathematica* su PPC sono tre le versioni di *Mathematica* acquistabili per Mac (ognuna delle quali contiene il sistema completo con tutte le possibilità).

1) La versione standard, compilata per il 68000 che non fa uso del coprocessore e gira su **tutti** i modelli Mac e PowerMac con sufficiente memoria. Questa è la versione che viene venduta a prezzo ultraridotto agli studenti.

2) La versione Enhanced, compilata per il 68020/30 che contiene sia una copia che non fa uso del coprocessore (per le macchine prive di FPU) che una copia che sfrutta tutte le possibilità delle macchine più potenti. Questa versione gira su tutte le macchine dal LC in su.

3) La versione compilata per PowerMac che ne sfrutta tutta la velocità e gira solo sui nuovi modelli.

Risultati dei test

Il test che ho passato è lo stesso presentato sul n. 136 di MC (Lepri e Tartarughe). Purtroppo (anzi per fortuna) le nuove macchine sono divenute così veloci che il test sta perdendo di significato come indicatore della velocità di elaborazione. Infatti alcune operazioni (in particolare il caricamento del pacchetto per l'elaborazione degli sviluppi in serie) sono rallentate dal "collo di bottiglia" degli accessi all'hard disk e costano fino al 50% dell'intera elaborazione. Per avere dati significativi senza cambiare benchmark (permettendo quindi a chi l'avesse acquisito di rifare le stesse prove) mi sono limitato a far passare il test due volte. La seconda volta i pacchetti sono stati caricati e il tempo misura in modo più significativo l'efficienza della CPU. Come termini di confronto ho usato un Quadra 800 e due 486 (ben forniti di RAM). La performance dei nuovi Mac è strabiliante (comparabile a quella di workstation che costano più del doppio). Si nota anche che vale sempre la pena di montare la memoria cache da 256K anche sul 6100 e il 7100 (il 6100 con cache va meglio del 7100 senza). Questo fatto indica che le potenzialità della CPU sono notevoli e non ancora tutte sfruttate dall'architettura e dal sistema operativo. La tabella mostra i tempi in secondi per la prima e la seconda esecuzione del test.

Macchina	*I passata*	*II Passata*
Quadra 800 68040 33 MHz	71.26	54.05
486/Dx 33 MHz 16Mbyte RAM	61.25	45.92
486/Dx2 66 MHz 16Mbyte RAM	36.86	27.74
Power 6100 60 MHz	36.40	18.45
Power 6100 60 MHz, 256K cache	30.40	13.16
Power 7100 66 MHz	28.93	13.56
Power 7100 66 MHz, 256K cache	25.88	11.18
Power 8100 80 MHz, 256K cache	22.55	9.53

Appena avrò nuovi dati ve li farò conoscere.

```
f[3,3,4]
```
Out[2]=
```
6
9
27
7625597484987
```
Un modo per seguire meglio quello che succede è quello di ridefinire **f** inserendo delle stampe. Ci viene in aiuto la funzione **HoldForm** che stampa i suoi argomenti senza valutarli. In altre parole **2+2** vale **4**, ma **HoldForm[2+2]** vale **2+2**.

In[3]:=
```
f[x_,y_,1]:=
   (Print[HoldForm[f[x,y,1]],"=",
          HoldForm[x+y]];
    x+y);
f[x_,1,z_]:=
   (Print[HoldForm[f[x,1,z]],"=",x];
    x)/;z>1;
f[x_,y_,z_]:=
 (Print[
    HoldForm[f[x,y,z]],"=",
    HoldForm[f[x,f[x,y1,z],z1]]/.
        {y1->y-1,z1->z-1}];
  f[x,f[x,y-1,z],z-1])
```
In[4]:=
```
f[3,3,3]
f[3,3,3]=f[3,f[3,2,3],2]
f[3,2,3]=f[3,f[3,1,3],2]
f[3,1,3]=3
f[3,3,2]=f[3,f[3,2,2],1]
f[3,2,2]=f[3,f[3,1,2],1]
f[3,1,2]=3
f[3,3,1]=3+3
f[3,6,1]=3+6
f[3,9,2]=f[3,f[3,8,2],1]
f[3,8,2]=f[3,f[3,7,2],1]
f[3,7,2]=f[3,f[3,6,2],1]
f[3,6,2]=f[3,f[3,5,2],1]
f[3,5,2]=f[3,f[3,4,2],1]
f[3,4,2]=f[3,f[3,3,2],1]
f[3,3,2]=f[3,f[3,2,2],1]
f[3,2,2]=f[3,f[3,1,2],1]
f[3,1,2]=3
f[3,3,1]=3+3
f[3,6,1]=3+6
f[3,9,1]=3+9
f[3,12,1]=3+12
f[3,15,1]=3+15
f[3,18,1]=3+18
f[3,21,1]=3+21
f[3,24,1]=3+24
```
Out[5]=
```
27
```
Per riuscire a seguire quello che succede per **f[3,3,4]** conviene ridefinire **f[x,y,2]** e **f[x,y,3]**.

In[6]:=
```
f[x_,y_,2]:=
    (Print[HoldForm[f[x,y,2]],"=",
           HoldForm[x y]];
           x y)
```
In[7]:=
```
f[x_,y_,3]:=
    (Print[HoldForm[f[x,y,3]],"=",
           HoldForm[x^y]];
     x^y)
```
In[8]:=
```
f[3,3,4]
f[3,3,4]=f[3,f[3,2,4],3]
f[3,2,4]=f[3,f[3,1,4],3]
f[3,1,4]=3
f[3,3,3]=3^3
f[3,27,3]=3^27
```
Out[8]=
```
7625597484987
```
In[9]:=
```
f[3,4,4]
f[3,4,4]=f[3,f[3,3,4],3]
f[3,3,4]=f[3,f[3,2,4],3]
f[3,2,4]=f[3,f[3,1,4],3]
f[3,1,4]=3
f[3,3,3]=3^3
f[3,27,3]=3^27
f[3,7625597484987,3]=3^7625597484987
```
Out[9]=
```
$Aborted
```
In[10]:=
```
f[4,3,4]
f[4, 3, 4]=f[4, f[4, 2, 4], 3]
f[4, 2, 4]=f[4, f[4, 1, 4], 3]
f[4, 1, 4]=4
f[4, 4, 3]=4^4
f[4, 256, 3]=4^256
```
Out[10]=
```
13407807929942597099574024998205\
 84612747936582059239337772356\
 44372176403007354697680187429\
 8166903427690031858186486050853\
 75388281194656994643364900608\
 4\
 096
```
In[11]:=
```
f[4,4,4]
f[4, 4, 4]=f[4, f[4, 3, 4], 3]
f[4, 3, 4]=f[4, f[4, 2, 4], 3]
f[4, 2, 4]=f[4, f[4, 1, 4], 3]
f[4, 1, 4]=4
f[4, 4, 3]=4^4
f[4, 256, 3]=4^256
f[4, 13407807929942597099574024\
  9820584612747936582059239337\
  7723561443721764030073546976\
  8018742981669034276900318581\
  8648605085375388281194656994\
  6433649006084096, 3]=
Power[4, 134078079299425970995\
  7402499820584612747936582059\
  2393377723561443721764030073\
```

```
  54697680187429816690342769OO\
  31858186486050853753882811941\
  65699464336490060840961
```

Out[11]=

```
$Aborted
```

Ed è meglio fermarsi qui!

Derivate e integrali calcolati una volta per tutte

È molto comodo usare *Mathematica* per calcolare simbolicamente gli integrali e derivate che poi servono nel seguito. È, però, chiaramente poco pratico rifare il calcolo ogni volta. Data una funzione calcoliamone le derivata nel punto 3.

In[1]:=

```
f[x_]=2 x^2 +4;
```

In[2]:=

```
D[f[x],x]
```

Out[2]=

```
4 x
```

In[3]:=

```
%/.x->3
```

Out[3]=

```
12
```

Dovendo utilizzare la derivata di f come fosse una funzione la seguente definizione NON VA BENE!

In[4]:=

```
df[x_]:=
  (Print["ora calcolo la derivata"];
   D[f[x],x]);
```

In[5]:=

```
df[3]
```

```
ora calcolo la derivata
```

Out[5]=

```
D[22, 3]
```

Una possibile strada è valutare **df** per un argomento generico **u** che poi viene sostituito con 3.

In[6]:=

```
df[u]/.u->3
```

```
ora calcolo la derivata
```

Out[6]=

```
12
```

La funzione **Evaluate** permette di valutare **df** una volta per tutte e di utilizzare il risultato nella definizione di una nuova funzione come se ci fossimo ricopiati il risultato simbolico. Si noti che la scritta "ora calcolo la derivata" non compare più quando si valuta **f1** perché la derivata simbolica è stata valutata una volta per tutte al momento della definizione di **f1**.

In[7]:=

```
f1[x_]:=Evaluate[df[x]]
```

```
ora calcolo la derivata
```

In[8]:=

```
f1[3]
```

Out[8]=

```
12
```

Lo stesso trucco permette di risparmiare un sacco di tempo nella valutazione di integrali simbolici.

In[9]:=

```
Timing[Integrate[Log[x],x]]
```

Out[9]=

```
{4.26667 Second, -x + x Log[x]}
```

In[10]:=

```
h[a_,b_]:=Simplify[
     Integrate[Log[x],{x,a,b}]]
```

In[11]:=

```
Timing[h[1,2]]
```

Out[11]=

```
{13.5333 Second, -1 + 2 Log[2]}
```

In[12]:=

```
g[a_,b_]:=Evaluate[Simplify[
     Integrate[Log[x],{x,a,b}]]]
```

In[13]:=

```
Timing[g[1,2]]
```

Out[13]=

```
{0.01 Second, -1 + 2 Log[2]}
```

Per finire ricordo come convenga usare **Evaluate** anche quando vengono plottate più funzioni in un solo grafico per accelerare il calcolo. Ecco il grafico di 5 polinomi di Chebyshev.

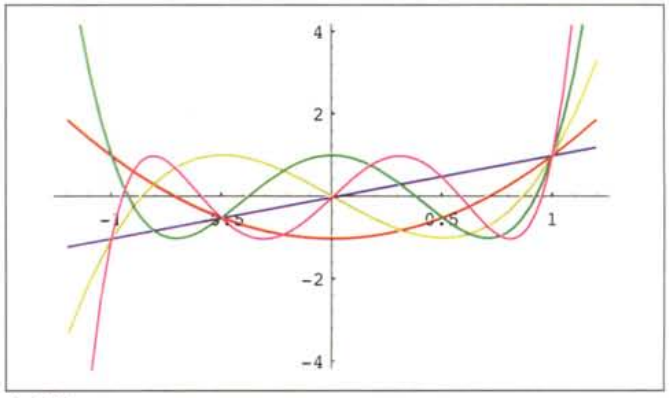
In[14]:=

```
<<Graphics`Colors`;
Timing[Plot[
 {ChebyshevT[1,x],
  ChebyshevT[2,x],
  ChebyshevT[3,x],
  ChebyshevT[4,x],
  ChebyshevT[5,x]},
     {x,-1.2,1.2},
      PlotStyle->
        {Blue,Red,Yellow,
         Green,Magenta}];]
```

Out[14]=

```
{3.3 Second, Null}
```

L'uso di **Evaluate[Table[...]]** dimezza il tempo del plottaggio della stessa figura.



In[15]:=

```
Timing[Plot[
  Evaluate[Table[
   ChebyshevT[n,x],{n,1,5}]],
     {x,-1.2,1.2},
      PlotStyle->
        {Blue,Red,Yellow,
         Green,Magenta}];]
```

Out[15]=

```
{1.36667 Second, Null)
```

MC

Francesco Romani è raggiungibile tramite Internet all'indirizzo romani@di.unipi.it

RISERVATO A CHI STA PENSANDO AL FUTURO

Per decidere l'acquisto di un computer
è necessario formulare **5 DOMANDE**

NELLA PRODUZIONE DI QUESTO COMPUTER SARANNO STATI UTILIZZATI **MATERIALI DI PRIMA QUALITÀ**?

IL COMPUTER CHE VOGLIO ACQUISTARE VIENE **COLLAUDATO** ALMENO **72 ORE** PRIMA DELLA CONSEGNA?

IL PUNTO VENDITA FA' PARTE DI UNA CATENA DI DISTRIBUZIONE PRESENTE SU TUTTO IL **TERRITORIO NAZIONALE**?

L'ASSISTENZA POST-VENDITA È GARANTITA DA **PERSONALE QUALIFICATO** CHE OPERA CON REGOLARE CERTIFICATO?

QUESTO PC HA UN GIUSTO **RAPPORTO QUALITÀ-PREZZO**?

LA RISPOSTA È UNA SOLA!

computer THOR

THOR computer

I NOSTRI MODELLI

- **386/SX 40**
- **386/DX 40**
- **486/DLC 40**
- **486/DX 40**
- **486/DX 33**
- **486/DX 50**
- **486/DX2-66**
- **PENTIUM 60**
- **PENTIUM 66**

E...ANCORA

NOTEBOOK
STAMPANTI
MONITOR
SCANNER
MULTIMEDIALE
MODEM
UPGRADE

LA NOSTRA OFFERTA

THOR PENTIUM 60

- *Case Big Tower display*
- *Motherboard INTEL PENTIUM 60 PCI*
- *8 Mb ram*
- *Hard Disk 270 Mb (2 anni di garanzia)*
- *Scheda Video Vga PCI S3 1MB esp. 2Mb*
- *Drive 3"1/2 1,44 Mb*
- *2 seriali, parallela*
- *Tastiera 102 tasti*
- *MS-DOS 6.2 + Windows for Workgroups*

3.998.000
iva esclusa

PUNTI VENDITA E AFFILIATI *THOR* COMPUTER

• **SISMAR INFORMATICA**	Via Vespucci 1 TORINO	☎ 011/58.19.007
• **SOFTEL**	Via Nizza 43 TORINO	☎ 011/66.90.962
• **COMPUTER LINE**	Via Chiampo 17 Pinerolo (TORINO)	☎ 0121/77.875
• **VALDATA COMPUTERS**	Via M.Grappa 4 AOSTA	☎ 0165/36.31.41
• **RA.VI.**	Via A.Moro 131 San Giorgio del Sannio (BENEVENTO)	☎ 0824/58.516
• **VGS INFORMATICA**	Via Fosse Ardeatine 100 FROSINONE	☎ 0775/21.21.88
• **ANDROMEDA SYSTEM**	Via Risorgimento 13 CANNETO LIPARI	☎ 090/98.12.525
• **INFORMATICA SICILIANA**	Via Kennedy 35/c CAPACI (PALERMO)	☎ 091/86.96.384
• **INFONET**	Via S. Domenico 39/b CASSANO M. (BARI)	☎ 080/77.63.09
• **GASPERINI**	Via Rosello 19 SASSARI	☎ 079/23.22.65
• **THOR COMPUTER FRANCE TECHNIWORK**	390,Chemin du Caladou VALBONNE (FRANCE)	☎ 33.93.12.10.12